Optical Font Family Recognition using a Neural Network

Drobac, Senka

Studia Humaniora Ouluensia
2019

Drobac , S & Lindén , K 2019 , Optical Font Family Recognition using a Neural Network . in J H Jantunen , S Brunni , N Kunnas , S Palviainen & K Västi (eds) , Proceedings of the Research Data And Humanities (Rdhum) 2019 Conference : Data, Methods And Tools . Studia humaniora ouluensia , no. 17 , Studia Humaniora Ouluensia , Oulu , pp. 115-123 .

http://hdl.handle.net/10138/306027

acceptedVersion

Optical Font Family Recognition using a Neural Network

Senka Drobac and Krister Lindén

1 Introduction

Working on OCR (Optical character recognition) of historical newspaper and journal data published in Finland, we found it beneficial to analyze and evaluate our OCR results based on font family. In Drobac et al., 2019 we indicate that recognizing the font family may be more important than recognizing the language of a document as a prerequisite for performing good OCR. This is a largely overlooked problem as many OCR tasks have specialized in documents with just one font, whereas in practice, historical newspapers were printed with several fonts. This is especially true during the transition period from Blackletter to Antiqua in Europe.

From a language technological point of view, the information on font family can be used to ensure that a sufficient amount of training data for machine learning to improve the quality of the OCR of historical newspapers is available. In addition, the font usage in periodicals can be used as to investigate the development of the printing press and the reading preferences during a transition period from 1800 until 1910 when Antiqua had largely replaced the Blackletter.

Our earlier data is mainly printed in Blackletter fonts and later data in Antiqua fonts, while in the transitioning period both font families were used at the same time, even on the same pages. Therefore, in order to make the recognition phase easier and faster, we are building one OCR model, which is able to recognize all fonts represented in the data. In order to make sure that we have sufficient training data for both font families, we need a font family classifier to simplify creation and sampling of training data.

Although there are existing tools for font classification, our problem seems to be overly specific. We only need to distinguish between Blackletter and the other fonts that were printed in Finland between 18th Century and early 20th Century, so the challenge is to find a simple enough font classifier for such a specific task.

Sahare et al., 2017 conveyed a detailed survey of different script identification algorithms. Zramdini et al., 1998 have developed a statistical approach of font recognition based on global typographical features. They report 97 % accuracy of font family recognition. Brodić et al., 2016 have approached a similar problem as we have, when they do identification of Fraktur and Latin scripts in German

historical documents using image texture analysis. The accuracy of their system has been reported to be 98.08 %.

In this work, we build a deep neural network binary font family classifier that for an image of one line of text decides whether it is written in Blackletter or Antiqua. Even with a simple configuration of the network, we get 97.5 % accuracy, leaving space for further improvement.

This font family classifier is specifically created for historical OCR for data printed in Finland. It is useful for collecting and analyzing the data, especially if the OCR is done with line-based software (Ocropy[1], Kraken[2], Calamary[3], Tesseract 4[4]). The font classifier is simple to use, in both the training and prediction phases. It is also easy to change network configurations and parameters.

[1] https://github.com/tmbdev/ocropy
[2] http://kraken.re/
[3] https://github.com/Calamari-OCR/calamari
[4] https://github.com/tesseract-ocr/tesseract/wiki/4.0-with-LSTM

2 Data and resources

In our experiments, we use three data sets: *img-lines, swe-6k* and *fin-7k*, all extracted from a corpus of historical newspapers and magazines that have been digitized by the National Library of Finland. Data ranges from 1771 until 1874 for *img-lines* and *swe-6k* sets, and from 1820 until 1939 for *fin-7k*.

a) b)

Fig. 1. Training examples: a) Antiqua, b) Blackletter

The first data set (*img-lines*) was created specifically for this task. The data set was collected from manually classified newspaper and journal pages. First, we randomly picked 307 pages from the entire corpus. We manually checked all the pages and classified them into three classes:

- *Antiqua-only* - pages that consist of mostly Antiqua font
- *Blackletter-only* - pages consist of mostly Blackletter font
- *Mixed pages* - pages with both Blackletter and Antiqua fonts

Then we segmented *Antiqua-only* and *Blackletter-only* pages into lines and cleaned the sets of poor quality, miss-segmented lines, or wrong font family. In the end, we were left with total of 6,356 Antiqua lines and 7,205 Blackletter lines.

Figure 1 shows five training examples from this set. On the left there are Antiqua lines and on the right Blackletter lines.

The other two data sets (*swe-6k* and *fin-7k*) were previously used for OCR of historical data. They both consist of randomly picked image lines, *swe-6k* has in total 6,158 image lines of Swedish text and *fin-7k* has 7,000 image lines of Finnish text. These two data sets had previously been divided into Blackletter and Antiqua and were used only for testing purposes.

Keras

Keras (Chollet Fet al., 2015) is a python library for machine learning. It runs on top of Tensorflow, and its simple and user friendly API (application programming interface) together with good quality documentation makes it easy to use.

In additions to neural networks, it also provides functions for image processing which allows dynamical augmentation of training data. This is particularly useful for image classification with small training sets, because in each iteration, the network receives a slightly altered image, and thus it never sees the same training image twice.

3 Method

In this section, we describe the preparation of the training and testing data and the structure of the neural network that we used to train the models. We also describe our evaluation methods.

3.1 Preparing the data

Although the original sizes of *img-lines* data sets were somewhat larger (Antiqua 6,356; Blackletter 7,205), we settled for 6,000 training lines from each category. Additionally, we reserved 200 lines for validation purpose and 100 lines for testing. As noted earlier, we used *swe-6k* and *fin-7*k exclusively for testing to have different test sets from the same corpus.

To load images into the neural network, we use a data generator with augmentation parameters on the training set, described in Table 2.

Table 2. Augmentation parameters used on training set

rescale=1./255	Rescaling factor
rotation_range=10	Degree range for random rotations.
width_shift_range=0.2	Fraction of total width
height_shift_range=0.2	Fraction of total height
shear_range=0.2	Shear Intensity
zoom_range=0.2	Range for random zoom. If a float,
	[lower, upper] = [1-zoom_range, 1+zoom_range]
fill_mode='nearest'	Points outside the boundaries of the input are filled according to the given
	mode: 'nearest': aaaaaaaa|abcd|ddddddd

It is also important to note that input image dimensions were set to 100x500 pixels for all training, validation and testing.

3.2 Neural network

The neural network consists of three pairs of convolution and pooling layers with a ReLU Activation function, followed by two fully connected layers and a Sigmoid

output layer that predicts the probability of the input image belonging to one of the two classes.

All convolution layers have a kernel size of 3 x 3 with zero padding. The first and the second layers have 32 filters and the third layer 64 filters. The pooling layers implement MaxPooling with a kernel size and stride of 2 x 2. Each fully connected layer has 64 hidden states, and the first layer has a ReLU Activation function. Between fully connected layers, we apply dropout (Srivastava et al., 2014) with a rate of 0.25 to prevent overfitting.

The loss is computed using Binary crossentropy with the RMSProp optimizer.

As an input for training, the algorithm expects a list of classified line images stored in respective file folders with corresponding names (i.e. Blackletter and Antiqua).

Figure 2 shows a diagram of this neural network.

3.3 Evaluation

For evaluation, we used the accuracy as a measure of correctly classified line images per total number of images, expressed in percentage:

$$accuracy = \frac{correct\ nomber}{total\ number} \cdot 100\%$$

To get the best model, we used early stopping on the validation set with patience 2, with the best weights restored.

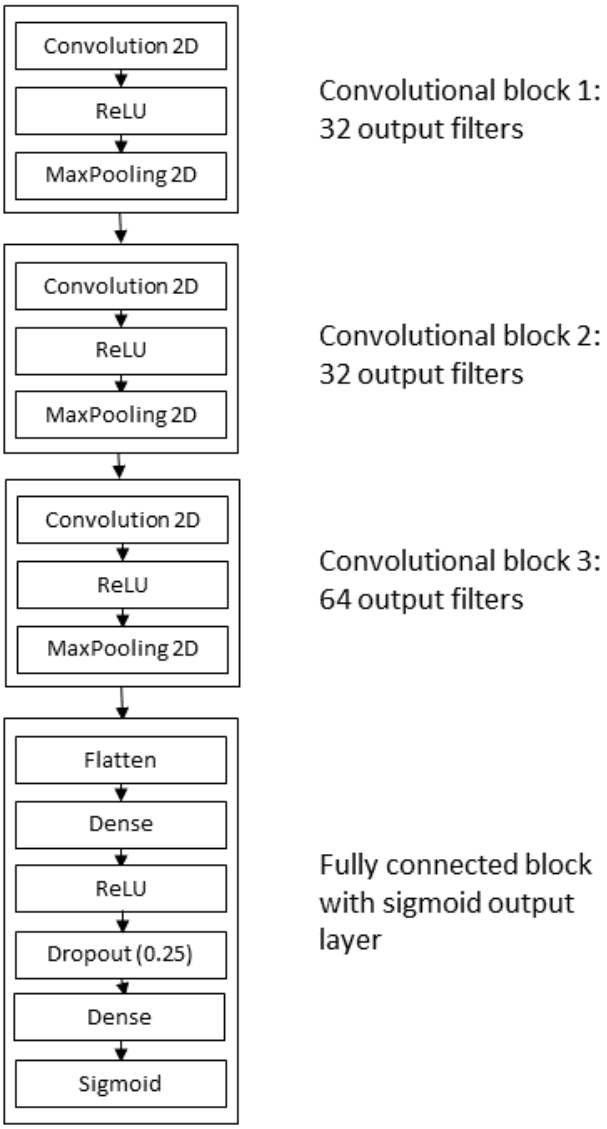

Fig. 2. Diagram of the neural network used for binary classification

On the unseen test sets, in addition to accuracy, we also calculate precision, recall and F_1 score. We define True Positive (TP) as correctly predicted Blackletter, False Positive (FP) as incorrectly predicted Blackletter, True Negative (TN) as correctly predicted Antiqua and False Negative (FN) as incorrectly predicted Antiqua. Then we calculate precision and recall as:

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

The F_1 score is defined as the measure that combines precision and recall using the harmonic mean:

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

4 Results

Table 3 shows prediction results on test sets swe-6k and fin-7k in terms of number of correctly and wrongly predicted line images.

Table 3. Prediction results on unseen data. On the left side, there are results on the test set swe-6k and on the right on fin-7k. The first row shows the number of predicted results for each font family category from a total of 6,158 test image lines in swe-6k and 7,000 images in fin-7k. The next two rows show how many were correctly and how many wrongly predicted.

Parameters	swe-6k		fin-7k	
	Antiqua	Blackletter	Antiqua	Blackletter
Predicted	2842	3316	1296	5704
True	2837	3054	1285	5540
False	5	262	11	164

Table 4 shows Accuracy of all three sets and Precision, Recall and F_1 score on *swe-6k* and *fin-7k*. Since *swe-6k* and *fin-7k* are used in a real world application (training of OCR models), we wanted to evaluate them further.

Table 4. Accuracy, Precision, Recall and F_1 score on different test sets. The first row shows results on the img-lines test set, the second on the swe-6k test set and the third on the fin-7k test set.

Test set	Accuracy	Precision	Recall	F_1
img-lines	97.5 %	-	-	-
swe-6k	95.64 %	99.8 %	92.1 %	95.8 %
fin-7k	97.5 %	99.8 %	97.1 %	98.4 %

5 Discussion and Conclusions

The accuracy results of 97.5 % on the classification test set and 95.64 % and 97.5 % on the real world test sets are quite high considering that we used only a basic setup without any experimentation with different model configurations or parameters. The high F_1 score shows that the model is quite good at balancing precision and recall, especially for *fin-7k*.

It is interesting to see that the accuracy on the *swe-6k* test set is almost 2 % lower than the accuracy on fin-7k, especially in light of the fact that this set is also more difficult for OCR than the Finnish set. Those two sets also differ on Antiqua and Blackletter ratio, with *swe-6k* having 46 % Antiqua and 54 % Blackletter fonts, while *fin-7k* only has 18 % Antiqua and 82 % Blackletter. It is possible that a larger number of Antiqua font lines in *swe-6k* also means a larger variety of fonts, making optical character recognition more challenging. It is possible that more training data and a deeper neural network would solve this problem.

Since this method is developed for a very specific task, it is difficult to make a comparison with other software without actually testing those systems on our data. However, comparing results that they get on their data with our results, we can see that we achieve a similarly high accuracy with a rather standard deep neural network approach.

This is a first approach to automatically classify our data to get enough training data from two font families, and for this purpose, we created a binary classifier. However, with simply changing the output layer to *softmax* and the loss function to *categorical crossentropy*, we could get a multi-class classifier with the same neural network setup.

List of references

Drobac, S., Kauppinen, P., & Linden, K. (Accepted/In press). Improving OCR of historical newspapers and journals published in Finland. In Proceedings of the 3nd International Conference on Digital Access to Textual Cultural Heritage ACM .

Sahare P & Dhok SB (2017) Script identification algorithms: a survey. International Journal of Multimedia Information Retrieval 6(3): 211–232.

Brodić D, Amelio A & Milivojević ZN (2016) Identification of fraktur and latin scripts in german historical documents using image texture analysis. Applied Artificial Intelligence 30(5): 379–395.

Zramdini A & Ingold R (1998) Optical font recognition using typographical features. IEEE Transactions on pattern analysis and machine intelligence 20(8): 877–882.

Chollet Fet al.(2015). Keras. https://keras.io.

Srivastava N, Hinton G, Krizhevsky A, Sutskever I & Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1): 1929–1958.